



ELSEVIER

Available online at www.sciencedirect.com

SCIENCE @ DIRECT®

Performance Evaluation 63 (2006) 1–14

**PERFORMANCE
EVALUATION**
An International
Journal

www.elsevier.com/locate/peva

Optimal processor allocation to differentiated job flows[☆]

Kimberly M. Wasserman^a, George Michailidis^{b,*}, Nicholas Bambos^c

^a *Mobile Wireless Group, Cisco Systems, Raleigh, NC 27604, USA*

^b *Department of Statistics, The University of Michigan, Ann Arbor, MI 48109-1092, USA*

^c *Department of Electrical Engineering and Management Science and Engineering,
Stanford University Stanford, CA 94305, USA*

Received 11 August 2003; received in revised form 19 October 2004

Available online 15 December 2004

Abstract

In this paper, we study the problem of dynamic allocation of the resources of a general parallel processing system, comprised of M heterogeneous processors and M heterogeneous traffic flows. Each traffic flow is modeled as a Bernoulli sequence of binary random variables, with the arrival rate depending on the job class. The service times of the jobs are independent random variables with distributions that depend on both the job class and the processor class. We characterize the stability region for this system and introduce a processor allocation policy that stabilizes the system under the maximum possible traffic loadings. We also investigate some special cases of this model and finally characterize its performance in the presence of finite buffers.

© 2004 Elsevier B.V. All rights reserved.

Keywords: Queueing; Multiprocessor system; Maximum throughput policies

1. Introduction

Consider a time-slotted queueing system consisting of M processors in parallel and M infinite capacity queues. There are M classes of job traffic. The class i jobs wait for processing in the i th queue and arrive according to the random process $\mathbf{A}_i = \{A_i(t); t \in \mathbb{Z}_+\}$, where $A_i(t) \in \{0, 1\}$ is the number of arrivals to

[☆] Earlier versions of the results were presented at the Applied Probability Conference in Ulm, 1999 and appeared in [14].

* Corresponding author. Tel.: +1 734 7633498; fax: +1 734 7634676.

E-mail address: gmichail@umich.edu (G. Michailidis).

the i th queue during the time slot $[t - 1, t)$, $i \in \mathbf{M} = \{1, 2, \dots, M\}$. It is further assumed that the $\{A_i\}_{i \in \mathbf{M}}$ processes are mutually independent sequences of Bernoulli random variables with $E[A_i(t)] = \lambda_i \in [0, 1]$. There are also M processors, whose service times are mutually independent random variables, independent of the arrival processes, with distributions that depend on both the job class and the processor class. The processing time of a class i job assigned to a class j processor has distribution P_{ji} and mean $0 < \mu_{ji}^{-1} < \infty$, $j, i \in \mathbf{M}$. Let $\mathcal{M} = \{\mu_{ji}\}$ denote the $M \times M$ service rate matrix, and $\vec{\lambda} = (\lambda_1, \lambda_2, \dots, \lambda_M)$ be the arrival rate vector.

At certain points in time, a processor must decide whether or not to process a job, and if so, from which of the M queues (job traffic classes). The decision mechanism employed by the processors at each of their decision points defines a processor allocation policy. If the decision points of all the processors occur at the same times, then the processors are said to be synchronous; otherwise they are asynchronous. We are interested in characterizing *dynamic* processor allocation policies that achieve maximal system throughput.

This basic queuing model captures the essence of a fundamental resource allocation problem encountered in many modern communication, computer, and manufacturing systems involving heterogeneous processors and multiple classes of job traffic flows. For example, consider the dynamic routing problem in a communication network implementing circuit-switching or wormhole routing [7]. Messages arrive to the router of a source node, distinguished by their destinations, and must be routed along one of a number of non-equivalent paths. For each source–destination pair, there are multiple paths through the network. The average transmission delay of a message depends on its destination *and* the path along which it is routed. Once the transmission of a message begins on a path, the path is not available to transmit other messages until this message has been completely received at the destination. The router faces the problem of allocating paths to messages. For example, if the preferred (fastest) path is not available the router may consider second best options.

As a second example, consider a heterogeneous distributed computing environment or a flexible manufacturing system or a call center consisting of a collection of processors with diverse computing or manufacturing or processing specialties and jobs with diverse processing requirements. Each processor is capable of processing any job, however the specialty of a processor determines its speed in processing each type of job. The objective of the scheduling policy is to dynamically assign processors to jobs so that maximal throughput for the system is achieved.

The subject of processor allocation in systems with parallel processors has received considerable attention in the literature. However, much of the work has concentrated on determining policies that optimize a specific cost function in the absence of external arrivals (see for example [16,15,9,2]). A special case of the system described above in which there is a single class of external arrivals and the performance measure of interest is the average number of customers in the system is studied in [12,4,6]. Finally, the issue of throughput has been addressed in [11] for a parallel processing system with activation constraints and in [1] for a parallel processing system with interacting resources.

In this paper, we are concerned with studying the manner in which processor allocation affects stability (throughput) *and* average number of customers. We consider a less complex topology than in [11], but permit much more general conditions on the processors and processing times, namely, the processors may be *asynchronous* and *non-preemptive*, and the processing times of the jobs depend on *both* the job class and the processor class, and are *not* required to be geometrically distributed. Our model also considers more flexible interactions between the incoming job

flows and the processors than in [1]. However, the stochastic framework in [1] is more general; the input job flows are assumed to be simply ergodic. This level of generality comes at a price of significantly weaker stability results (rate ergodicity of the departure process, which does not necessarily imply finite backlogs).

The main contributions of the paper are as follows: we first provide a *complete* characterization of the stability region, specifying a necessary and sufficient condition for stability in terms of the input and service rates $\bar{\lambda}$ and \mathcal{M} , respectively. The necessity of the stability condition is established using a coupling argument [5]. As for sufficiency, we consider a simple dynamic processor allocation policy π_{MWQ} , which allocates a processor to the queue with the largest weighted queue length, where the weights are given by the elements of the service rate matrix \mathcal{M} . In the case of synchronous and preemptive processors, we use drift analysis [3] to show that π_{MWQ} stabilizes the system, under a natural condition on the processing time distributions, whenever the stability condition holds. The policy π_{MWQ} therefore achieves maximal throughput. The result extends to the case of asynchronous and non-preemptive processors, if the processing time distributions are geometric. We then consider a partially-symmetric system where $P_{ji} = P_c$, $j \neq i$, and $\mu_c \leq \mu_{jj}$, $j, i \in \mathbf{M}$, and consider another dynamic policy π_{MSR} , which allocates a class j processor to a class j job if one is present, and to a job from the queue with the greatest number of jobs otherwise, $j \in \mathbf{M}$. Using a coupling argument, it is shown that π_{MSR} not only outperforms π_{MWQ} with respect to average number of customers in the system, but in the totally symmetric system, where $\lambda_j = \lambda$ and $P_{jj} = P$, $j \in \mathbf{M}$, achieves minimal average system size. Finally, by exploiting the structure of π_{MSR} we obtain a simple bound on the average system size under π_{MWQ} .

The paper is organized as follows. In Section 2, the system dynamics are described. The stability results appear in Section 3 and the average system size results appear in Section 4. Some concluding remarks are drawn in Section 5.

2. System evolution

Let Γ denote the set of all non-idling processor allocation policies. For an arbitrary policy $\gamma \in \Gamma$, define the k th allocation sequence generated by γ as $\mathbf{U}_k^\gamma = \{U_k^\gamma(t) = (U_{k1}^\gamma(t), U_{k2}^\gamma(t), \dots, U_{kM}^\gamma(t)); t \in \mathbb{Z}_+\}$, where $U_{ki}^\gamma(t) \in \{0, 1\}$ is equal to one only if the k th processor is allocated to a class i job during the time slot $[t, t + 1)$, $k \in \mathbf{M}$. Each processor may process at most one job during a time slot and so $\mathbf{U}_k^\gamma(t) \in \{0, e_1, e_2, \dots, e_M\}$, where e_i is an M -dimensional basis vector with the i th element equal to one and the remaining elements equal to zero, and 0 is an M -dimensional vector with all elements equal to zero. Let $\mathbf{U}^\gamma = (\mathbf{U}_1^\gamma, \mathbf{U}_2^\gamma, \dots, \mathbf{U}_M^\gamma)$ be the allocation sequence generated by policy γ . Also define the k th processing sequence generated by γ as $\mathbf{S}_k^\gamma = \{S_k^\gamma(t); t \in \mathbb{Z}_+\}$, where $S_k^\gamma(t) \in \{0, 1\}$ is equal to one only if the k th processor completes the processing of a job during $[t - 1, t)$. Lastly, define the queue length process generated by γ as $\mathbf{N}^\gamma = \{N^\gamma(t) = (N_1^\gamma(t), N_2^\gamma(t), \dots, N_M^\gamma(t)); t \in \mathbb{Z}_+\}$, where $N_i^\gamma(t) \in \mathbb{Z}_+$ is the number of jobs in the i th queue, that is, the number of class i jobs waiting to be processed, at time t .

We restrict attention to the set of non-anticipative and stationary processor allocation policies $\Pi \subset \Gamma$. A policy $\pi \in \Pi$ is a function of only the queue lengths and the processor class, taking values in the set $\{0, e_1, e_2, \dots, e_M\}$, that is, $\pi : \mathbb{Z}_+^M \times \mathbf{M} \rightarrow \{0, e_1, e_2, \dots, e_M\}$. For simplicity, we assume that if at some time t , more processors are allocated to a queue than there are jobs in that queue, then a number of

processors equal to the surplus are randomly chosen to idle during $[t, t + 1)$. For arbitrary $\pi \in \mathbf{\Pi}$, the i th queue length evolves according to

$$N_i^\pi(t + 1) = N_i^\pi(t) + A_i(t + 1) - \sum_{k=1}^M U_{ki}^\pi(t) S_k^\pi(t + 1), \quad (2.1)$$

for $t \geq 0$, $i = 1, 2, \dots, M$.

3. System stability issues

In this section, we address the issue of system stability. For a given service rate matrix \mathcal{M} and arrival rate vector $\vec{\lambda}$, the system is said to be *stable* under $\pi \in \mathbf{\Pi}$, if the conditional expectation $E[N^\pi(t) | N^\pi(0)]$ is uniformly bounded above by a constant. The set of arrival rate vectors for which the system is stable under π is denoted by $\mathbf{\Lambda}_\pi$.

Let \mathcal{F} be the set of all $M \times M$ stochastic matrices $f = \{f_{ij}\}$ such that $f_{ij} \in [0, 1]$ for $i, j \in \mathbf{M}$ and $\sum_{j=1}^M f_{ij} = 1$. Define the set

$$\mathbf{\Lambda} = \left\{ \vec{\lambda} \in [0, 1]^M : \exists f \in \mathcal{F} \text{ for which } \sum_{i=1}^M \frac{\lambda_i f_{ij}}{\mu_{ji}} < 1, \forall j \in \mathbf{M} \right\}. \quad (3.1)$$

Intuitively the above condition says that the sum of “normalized” (by the corresponding processing rates) flows to be served by processor j must be less than 1 (the nominal “normalized” capacity of the processor).

Consider next the Maximum Weighted Queue-Length (MWQ) policy $\pi_{\text{MWQ}} \in \mathbf{\Pi}$ defined as

$$\pi_{\text{MWQ}}(z, j) = \begin{cases} e_m & \text{if } z_m \mu_{jm} = \max_i \{z_i \mu_{ji}\}, \\ 0 & \text{if } z_i = 0, \forall i, \end{cases}$$

with $z \in \mathbb{Z}_+^M$ and $j \in \mathbf{M}$. If z_i is the number of class i jobs in the i th queue at some time slot t , then the product $z_i \mu_{ji}$ is a measure of how much work there is in the i th queue at t as seen by a class $j \in \mathbf{M}$ processor. The policy π_{MWQ} is then, attempting to balance weighted workloads. Moreover, when the differences between the queue lengths are small, then π_{MWQ} acts like an “index” policy, allocating processors to the queues where they are the most efficient. However, if the number of jobs in a particular queue is rapidly accumulating, then π_{MWQ} acts like a “longest queue” policy, allocating more and more processors to the “runaway” queue. Note that π_{MWQ} does *not* require knowledge of the arrival rate vector $\vec{\lambda}$.

In order to show that the proposed π_{MWQ} policy stabilizes the system under the maximum possible throughput (i.e. $\vec{\lambda} \in \mathbf{\Lambda}$), we make use of the following Fact. Recall, if $\mathbf{Z} = \{Z(t); t \geq 0\}$ and $\mathbf{Y} = \{Y(t); t \geq 0\}$ are real-valued random processes, then $(X(t)|Z(t)) \leq_{st} Y(t)$ if $P\{X(t) > a | Z(t)\} \leq P\{Y(t) > a\}$ for all $a \in \mathbb{R}$ and $t \geq 0$ (see [5]).

Fact (see Hajek [3]). Suppose $\mathbf{N} = \{N(t), t \in \mathbb{Z}_+\}$ is an irreducible and aperiodic Markov chain with countable state space \mathcal{E} . If, for some $-\infty \leq a < \infty$ and $\epsilon > 0$, there exists a lower bounded, real-valued (Lyapunov) function $V : \mathcal{E} \rightarrow \mathbb{R}$ such that

- (C1) $E[V(N(t + 1)) - V(N(t)) \mid N(t) = z] \leq -\epsilon$ for all $z \in \{x \in \mathcal{E} : V(x) > a\}$ and $t \geq 0$.
- (C2) There exists a real-valued random variable Z such that $(|V(N(t + 1)) - V(N(t))| \mid N(t)) \leq_{st} Z$ for all $t \geq 0$ and $E[e^{\nu Z}] = D$ is finite for some $\nu > 0$.

Then $E[V(N(t)) \mid N(0)]$ is uniformly bounded above by a constant.

It is important to note that the set $\{x \in \mathcal{E} : V(x) \leq a\}$ is *not* required to be finite. The intuition behind this fact is that when the underlying state of the queueing system becomes 'large', there exists a negative feedback mechanism (the negative drift condition C1) that pulls the system towards the empty state. The second condition (C2) imposes a constraint on the increments process of $V(N(t))$, necessary for the result to hold (see also [8]).

3.1. Sufficiency of Λ : synchronous and preemptive processors

In this section, we assume that (1) the processors are synchronous (decision times coincide for all M processors) and preemptive (processors are allowed to stop processing a particular flow and can be allocated to another flow), and (2) that the processing time distribution P_{ji} is geometric with parameter μ_{ji} , $j, i \in \mathbf{M}$. Under these assumptions, it is easy to see that $\mathbf{N}^{\pi_{\text{MWQ}}}$ is a Markov chain.

Proposition 3.1. *If $\vec{\lambda} \in \Lambda$ then the system is stable under π_{MWQ} .*

Proof. Fix the input vector $\vec{\lambda} \in \Lambda$. Then there exist $\{f_{ij}\}$ such that $\sum_{i=1}^M (\lambda_i f_{ij} / \mu_{ji}) - 1 < 0$, for $j \in \mathbf{M}$. We will prove the stability of the system under the MWQ policy by showing that both conditions of the Fact are satisfied. Define the Lyapunov function $V : \mathbb{Z}_+^M \rightarrow \mathbb{R}$ as $V(z) = (\sum_{i=1}^M z_i^2)^{1/2} = \|z\|$. Since there can be at most one arrival to each queue and at most M departures from the system during a time slot, the second condition (C2) of the Fact is trivially satisfied. As for condition (C1), consider first the Lyapunov function $\tilde{V}(z) = \|z\|^2$. Suppose

$$E[\|N(t + 1)\|^2 \mid N(t) = z] \leq \|z\|^2 - \eta \sum_{i=1}^M z_i + \beta \tag{3.2}$$

for some constants $\eta > 0$ and $\beta > 0$, where we have suppressed the dependence on π_{MWQ} from the notation. Then there exists $\epsilon > 0$ such that

$$\begin{aligned} E[\|N(t + 1)\| \mid N(t) = z] &\leq (E[\|N(t + 1)\|^2 \mid N(t) = z])^{1/2} \\ &\leq \left(\|z\|^2 - \eta \sum_{i=1}^M z_i + \beta \right)^{1/2} \leq \|z\| - \epsilon \end{aligned} \tag{3.3}$$

for $\sum_{i=1}^M z_i$ large enough, and so condition (C1) holds. We must therefore show that (3) holds. Define $\Delta_i(t + 1) = A_i(t + 1) - \sum_{k=1}^M U_{ki}(t)S_k(t + 1)$, $i \in \mathbf{M}$. We then have

$$\begin{aligned}
E[\|N(t+1)\|^2 \mid N(t) = z] - \|z\|^2 &= E[\|\Delta_i(t+1)\|^2 \mid N(t) = z] \\
&\quad + 2E \left[\sum_{i=1}^M N_i(t) \Delta_i(t+1) \mid N(t) = z \right] \\
&\leq \beta + 2 \left(\sum_{i=1}^M z_i \lambda_i - \sum_{i=1}^M z_i \sum_{j=1}^M U_{ji}(t) \mu_{ji} \right) \\
&= \beta + 2 \left(\sum_{j=1}^M \sum_{i=1}^M z_i \mu_{ji} \frac{\lambda_i f_{ij}}{\mu_{ji}} - \sum_{j=1}^M \max_i \{z_i \mu_{ji}\} \right) \\
&\leq \beta + 2 \left(\sum_{j=1}^M \max_i \{z_i \mu_{ji}\} \left(\sum_{i=1}^M \frac{\lambda_i f_{ij}}{\mu_{ji}} - 1 \right) \right) \\
&\leq \beta - \eta \sum_{i=1}^M z_i, \tag{3.4}
\end{aligned}$$

where the last equality of the second line of (5) follows from the definition of the policy π_{MWQ} and the last inequality follows because $\min_{z_i \geq 0; i=1, \dots, M} \sum_{i=1}^M z_i / (\sum_{i=1}^M z_i^2) = 1$. Therefore, if $\vec{\lambda} \in \mathbf{\Lambda}$ then the system is stable under π_{MWQ} by the Fact and $\mathbf{\Lambda}_{\pi_{\text{MWQ}}} = \mathbf{\Lambda}$; that is the set of arrival vectors $\vec{\lambda}$, $\mathbf{\Lambda}_{\pi_{\text{MWQ}}}$ that under the MWQ policy lead to finite backlogs for all the queues in the system coincides with the maximal set of arrival vectors $\mathbf{\Lambda}$ that the system's resources can accommodate. Hence, we can claim that the MWQ policy *maximizes* the stability region of the system. \square

3.2. Sufficiency of $\mathbf{\Lambda}$: asynchronous and non-preemptive processors

In this section, we relax the assumption of pre-emptive processors and assume instead that (1) the processors are asynchronous, non-preemptive, and non-idling, and (2) the following natural condition on the processing time distributions P_{ji} holds, $j, i \in \mathbf{M}$.

Condition 1. Let X_{ji} be a random variable with distribution P_{ji} . Then there exist constants $L_{ji} < \infty$ and $\nu_{ji} \in (0, 1]$ such that

$$\sup_t \{P(X_{ji} - t \leq L_{ji} \mid X_{ji} > t)\} \geq \nu_{ji}, \quad i, j \in \mathbf{M}. \tag{3.5}$$

The above condition says that no matter how long a particular processor has been serving a particular job, there is a chance of at least $\min_{j,i} \{\nu_{ji}\}$ that service will be completed within the next $\max_{j,i} \{L_{j,i}\}$ time slots.

For arbitrary $\pi \in \mathbf{\Pi}$, define $\beta^\pi = \{\beta^\pi(t) = \beta_1^\pi(t), \beta_2^\pi(t), \dots, \beta_M^\pi(t); t \in \mathbb{Z}_+\}$, where $\beta_k^\pi(t) \in \{0, 1, \dots, t\}$ is the number of time slots since the last service completion before t by the k th processor, $k \in \mathbf{M}$. Define the random process $\Phi^\pi = \{\Phi^\pi(t) = (N^\pi(t), \beta^\pi(t), U^\pi(t)); t \in \mathbb{Z}_+\}$. Under the above assumptions, Φ^π is a Markov chain.

Proposition 3.2. If $\vec{\lambda} \in \mathbf{\Lambda}$ then the system is stable under π_{MWQ} .

Proof. Fix the input vector $\vec{\lambda} \in \mathbf{\Lambda}$. Define $\delta_j = 1 - \sum_{i=1}^M \lambda_i f_{ij} / \mu_{ji}$, $j \in \mathbf{M}$, and let $\delta_* = \min_j \{\delta_j\}$. Since Condition 1 above holds, and π_{MWQ} is stationary and non-idling, there exists $R < \infty$ such that

$$E \left[\sum_{i=1}^M \sum_{s=t}^{t+R-1} \frac{U_{ki}^{\pi_{\text{MWQ}}}(s) S_k^{\pi_{\text{MWQ}}}(s+1)}{\mu_{ki}} \mid \phi(t) = (z, b, u) \right] \geq R \left(1 - \frac{\delta_*}{2M} \right) \quad (3.6)$$

for all $t \geq 0$, by the Elementary Renewal Theorem [10].

We will again prove stability by using the Fact. Define the Lyapunov function $V : \mathbb{Z}_+^M \times \mathbb{Z}_+^M \times \{0, e_1, e_2, \dots, e_M\}^M \rightarrow \mathbb{R}$ as $V((z, b, u)) = \|z\|$. Condition (C2) of the Fact is again trivially satisfied, so we need only to show that condition (C1) holds. It is enough to show there exists $\epsilon > 0$ such that

$$E[\|N(t+R)\| - \|N(t)\| \mid \phi(t) = (z, b, u)] \leq -\epsilon \quad (3.7)$$

for $\sum_{i=1}^M z_i$ large enough, where we have again suppressed the dependence on the policy from the notation.

Now, suppose the k th processor completes the processing of a job during the time slot $[t+s-1, t+s)$, and is allocated to the b th queue at time $t+s$, for $k, b \in \mathbf{M}$. From the definition of the MWQ policy, we must then have $N_b(t+s)\mu_{kb} \geq N_i(t+s)\mu_{ki}$, for $i \in \mathbf{M}$. From (2.1), $N_b(t+s) \leq N_b(t) + s$ and $N_i(t+s) \geq N_i(t) - Ms$. Therefore,

$$\max_i \{N_i(t)\mu_{ki}\} - N_b(t)\mu_{kb} \leq 2M\mu^*s, \quad (3.8)$$

where $\mu^* = \max_{ji} \{\mu_{ji}\}$.

Consider the Lyapunov function $\tilde{V}(z, b, u) = \|z\|^2$. As in the proof of Proposition 3.1, in order to show that (3.7) holds, we need only show that

$$E[\|N(t+R)\|^2 \mid \phi(t) = (z, b, u)] \leq \|z\|^2 - \hat{\eta}R \sum_{i=1}^M z_i + \hat{\beta} \quad (3.9)$$

for some constants $\hat{\eta} > 0$ and $\hat{\beta} > 0$. We have that

$$\begin{aligned} & E[\|N(t+R)\|^2 - \|N(t)\|^2 \mid \phi(t) = (z, b, u)] \\ & \leq \hat{\beta} + 2E \left[\sum_{i=1}^M \sum_{s=t}^{t+R-1} (N_i(s)(A_i(s+1) - \sum_{k=1}^M U_{ki}(s)S_k(s+1))) \mid \phi(t) = (z, b, u) \right]. \end{aligned} \quad (3.10)$$

Using first (3.8) and then (3.6), we have

$$\begin{aligned} & E \left[\sum_{i=1}^M \sum_{s=t}^{t+R-1} N_i(t) \left(\sum_{k=1}^M U_{ki}(s)S_k(s+1) \right) \mid \phi(t) = (z, b, u) \right] \\ & \geq E \left[\sum_{i=1}^M \sum_{s=t}^{t+R-1} \sum_{k=1}^M \left(\max_i \{z_i \mu_{ki}\} - 2M\mu^*(s-t) \right) \frac{U_{ki}(s)S_k(s+1)}{\mu_{ki}} \mid \phi(t) = (z, b, u) \right] \\ & \geq R \sum_{j=1}^M \max_i \{z_i \mu_{ji}\} \left(1 - \frac{\delta_*}{2M} \right) - \frac{2M^2\mu^*}{\mu_*} R(R+1). \end{aligned} \quad (3.11)$$

Substituting (3.11) into (3.9) and proceeding along similar lines as in the proof of Proposition 3.1 establishes (3.8). Therefore, if $\vec{\lambda} \in \mathbf{\Lambda}$ then the system is stable under π_{MWQ} by the Fact and $\mathbf{\Lambda}_{\pi_{\text{MWQ}}} = \mathbf{\Lambda}$. \square

3.3. Necessity of Λ

In the following proposition, we use a coupling argument to show that $\vec{\lambda} \in \Lambda$ is a necessary condition for stability under any policy in Π . This proposition shows that if $\vec{\lambda} \notin \Lambda$ no control policy in Π can keep the backlogs finite.

Proposition 3.3. *If the system is stable under a policy $\pi \in \Pi$, then $\vec{\lambda} \in \Lambda$.*

Proof. We establish the result for the case of synchronous and preemptive processors, and geometric processing times with parameters μ_{ji} , $j, i \in \mathbf{M}$. The proof is easily extended to the case where the processors are asynchronous and non-preemptive, and the processing time distributions P_{ji} are arbitrary.

Fix $\vec{\lambda} \in [0, 1]^M$. Since the system is stable under π , \mathbf{N}^π is an irreducible and positive recurrent Markov chain, which may be taken to be stationary. We must then have

$$\lambda_i = \mu_{1i}E[U_{1i}^\pi(t)] + \mu_{2i}E[U_{2i}^\pi(t)] + \cdots + \mu_{ki}E[U_{ki}^\pi(t)] + \cdots + \mu_{Mi}E[U_{Mi}^\pi(t)] \quad (3.12)$$

for $i \in \mathbf{M}$. Define $f_{ij} = \mu_{ji}E[U_{ji}^\pi(t)]/\lambda_i$. From (3.12), we have $\sum_{j=1}^M f_{ij} = \sum_{j=1}^M \mu_{ji}E[U_{ji}^\pi(t)]/\lambda_i$, $i \in \mathbf{M}$, and so $f \in \mathcal{F}$. Since a processor can only serve one job at a time, we also have that $\sum_{i=1}^M \lambda_i f_{ij}/\mu_{ji} = \sum_{i=1}^M E[U_{ji}^\pi(t)] \leq 1$, for $j \in \mathbf{M}$. Therefore $\vec{\lambda} \in \bar{\Lambda}$, where

$$\bar{\Lambda} = \left\{ \lambda \in [0, 1]^M : \exists f \in \mathcal{F} \text{ for which } \sum_{i=1}^M \frac{\lambda_i f_{ij}}{\mu_{ji}} \leq 1 \quad \forall j \in \mathbf{M} \right\}. \quad (3.13)$$

Now since \mathbf{N}^π is positive recurrent, there exists $\delta > 0$ such that $P(\sum_{i=1}^M N_i^\pi(t) = 0) \geq \delta$. Pick $\epsilon_i > 0$, $i \in \mathbf{M}$, such that $\sum_{i=1}^M \epsilon_i < \delta\mu_*$, where $\mu_* = \min_{ji}\{\mu_{ji} : \mu_{ji} > 0\}$. Increase the arrival rate to the i th queue by ϵ_i , so that the total arrival rate to the i th queue is $\lambda'_i = \lambda_i + \epsilon_i$. Color an arriving class i job “red” with probability λ_i/λ'_i and “blue” with probability ϵ_i/λ'_i . Let policy π give priority to the red jobs, and serve blue jobs only when the system is completely empty. By hypothesis, the conditional expected number of red jobs is uniformly bounded. Since $\sum_{i=1}^M \epsilon_i < \delta\mu_*$, the conditional expected number of blue jobs must also be uniformly bounded (using the Fact). Therefore, it follows that $\vec{\lambda}' \in \bar{\Lambda}$ and since all $\epsilon_i > 0$ that $\vec{\lambda} \in \Lambda$. \square

Remark. It should be noted that all the results go through if we generalize the model to have a number $c_j \geq 1$ of type j processors. The stability region Λ is then characterized by the set of inequalities $\sum_{i=1}^M \lambda_i f_{ij}/\mu_{ji} < c_j$, $\forall j \in \mathbf{M}$.

3.4. Partially-symmetric systems

In this section, we restrict attention to the case where $P_{ji} = P_c$, $j \neq i$, and $\mu_{jj} \geq \mu_c$, $j, i \in \mathbf{M}$. Thus the class j processor is most efficient when serving class j jobs. For $\vec{\lambda} \in [0, 1]^M$, let $\mathcal{B}_{\vec{\lambda}} = \{i \in \mathbf{M} : \lambda_i \geq \mu_{ii}\}$. In this case the stability region is given by

$$\Lambda = \left\{ \lambda \in [0, 1]^M : \sum_{i \in \mathcal{B}_{\vec{\lambda}}} \frac{\lambda_i - \mu_{ii}}{\mu_c} < \sum_{i \notin \mathcal{B}_{\vec{\lambda}}} \left(1 - \frac{\lambda_i}{\mu_{ii}} \right) \right\}. \quad (3.14)$$

Intuitively this condition says that the extra capacity available in the system from the processors *not in* $\mathcal{B}_{\vec{\lambda}}$ exceeds the excess demand from the job classes in $\mathcal{B}_{\vec{\lambda}}$.

Now consider the Maximum Service Rate (MSR) policy $\pi_{\text{MSR}} \in \Pi$ defined by

$$\pi_{\text{MSR}}(z, j) = \begin{cases} e_j & \text{if } z_j > 0, \\ e_m & \text{if } z_j = 0 \text{ and } z_m = \max_i \{z_i\}, \\ 0 & \text{if } z_i = 0 \forall i, \end{cases}$$

$z \in \mathbb{Z}_+^M$ and $j \in \mathbf{M}$. Under the MSR policy, a class j processor is allocated to a class j job if one is present; otherwise, it is allocated to a job from the queue with the greatest number of jobs. In other words, the policy MSR policy allocates a processor according to π_{MWQ} , only if the queue where it is the most efficient is empty. Note that this policy *also* does *not* require knowledge of the arrival rate vector $\vec{\lambda}$ or the service rate matrix \mathcal{M} . In the following proposition, we use a coupling argument to show that π_{MSR} achieves maximal throughput.

Proposition 3.4. *If $\vec{\lambda} \in \Lambda$ then the system is stable under π_{MSR} .*

Proof. We show the case in which the processors are synchronous and preemptive, and the processing time distribution P_{ji} is geometric with parameter μ_{ji} , $j, i \in \mathbf{M}$. The proof easily extends to the case where the processors are asynchronous and non-preemptive, and the processing time distributions P_{ji} are arbitrary.

Fix $\vec{\lambda} \in \Lambda$. Let $\mathcal{G} = \{i \in \mathbf{M} : \lambda_i < \mu_{ii}\}$ and $\mathcal{B} = \{i \in \mathbf{M} : \lambda_i \geq \mu_{ii}\}$. Since $\vec{\lambda} \in \Lambda$, there exist $\{f_{ij}\} \in \mathcal{F}$ such that $f_{ii} + \sum_{j \in \mathcal{G}} f_{ij} = 1$ for all traffic flows $i \in \mathcal{B}$, and such that $\lambda_j f_{jj} < \mu_{jj}$ for all processors $j \in \mathcal{B}$ and $\sum_{i \in \mathcal{B}} f_{ij} \lambda_i / \mu_c + \lambda_j / \mu_{jj} < 1$ for all processors $j \in \mathcal{G}$. Consider the policy $\pi_r \in \Pi$ defined as follows. If traffic flow $i \in \mathcal{G}$, then each class i arrival is routed to the i th queue. If traffic flow $i \in \mathcal{B}$, then a class i arrival is routed to the i th queue with probability f_{ii} , and to the j th queue with probability f_{ij} , with $j \in \mathcal{G}$. On the processor side, the class $j \in \mathbf{M}$ processors only serve the jobs that have been routed to the j th queue, and give priority to the class j jobs. Under π_r , if the i th queue is empty at time $t \geq 0$, then a class i processor with $i \in \mathcal{G}$ may only serve a class \mathcal{B} job.

Couple the realizations [5] of the queue length processes under policies π_r and π_{MSR} by giving the i th queue in each system the same arrival process, $i \in \mathbf{M}$, and giving the j th processor in each system the same service outcomes during all time slots except those slots during which one of these processors is allocated to the j th queue and the other is not, $j \in \mathbf{M}$. During the intervals of time in which

$$\max_{i \in \mathcal{G}} \{N_i^{\pi_{\text{MSR}}}(t)\} \geq \max_{i \in \mathcal{B}} \{N_i^{\pi_{\text{MSR}}}(t)\}, \tag{3.15}$$

the number of class \mathcal{B} jobs under π_{MSR} is less than M times the number of \mathcal{G} jobs under π_{MSR} . During the intervals of time in which

$$\max_{i \in \mathcal{G}} \{N_i^{\pi_{\text{MSR}}}(t)\} < \max_{i \in \mathcal{B}} \{N_i^{\pi_{\text{MSR}}}(t)\}, \tag{3.16}$$

whenever a class \mathcal{G} queue under π_{MSR} empties, the ‘‘associated’’ servers are allocated to class \mathcal{B} queues, that is, the servers under π_{MSR} act like those under π_r . Moreover, the difference between the number of class \mathcal{B} jobs under π_{MSR} and π_r during such an interval is never greater than their difference at the

beginning of that interval, which is in turn less than the number of class \mathcal{G} jobs under π_{MSR} at the beginning of the interval. Since the individual queues in \mathcal{G} are stable under both π_{MSR} and π_r , if the class \mathcal{B} queues under π_{MSR} are not stable, then neither are the class \mathcal{B} queues under π_r , which is a contradiction since $\lambda_j f_{jj} < \mu_{jj}$ for all $j \in \mathcal{B}$ and $\sum_{i \in \mathcal{B}} f_{ij} \lambda_i / \mu_c + \lambda_j / \mu_{jj} < 1$ for all $j \in \mathcal{G}$. We then conclude $\vec{\lambda} \in \Lambda_{\pi_{\text{MSR}}}$. \square

Remark. In the general case, a policy $\bar{\pi} \in \Pi$ analogous to π_{MSR} , which allocates a processor according to π_{MWQ} only if the queue where it is the most efficient is empty, does not generally achieve maximal throughput. Indeed, consider a system consisting of three queues and three processors. Let the service rate matrix be

$$\mathcal{M} = \begin{bmatrix} \alpha & 0 & 0 \\ \epsilon & \alpha & \alpha \\ \beta & \alpha & \alpha \end{bmatrix}.$$

Assume $\lambda_2 + \lambda_3 < \alpha$, and $\epsilon < \beta < \alpha$. Under π_{MWQ} , a necessary condition for stability is

$$\lambda_1 < \alpha + \beta + \epsilon \left(1 - \frac{\lambda_2 + \lambda_3}{\alpha} \right) = \lambda^*.$$

Now consider the policy $\bar{\pi} \in \Pi$ defined by

$$\bar{\pi}(z, j) = \begin{cases} e_j & \text{if } z_j > 0, \\ e_m & \text{if } z_j = 0 \text{ and } z_m \mu_{jm} = \max_i \{z_i \mu_{ji}\}, \\ 0 & \text{if } z_i = 0, \forall i, \end{cases}$$

with $z \in \mathbb{Z}_+^M$ and $j \in \mathbf{M}$. Under $\bar{\pi}$, a necessary condition for stability is

$$\lambda_1 < \alpha + \epsilon \left(1 - \frac{\lambda_2}{\alpha} \right) + \beta \left(1 - \frac{\lambda_3}{\alpha} \right) < \lambda^*.$$

4. Average system size issues

In this section, we address the issue of average system size. However, we need to restrict attention to the partially symmetric system of the previous subsection. In addition, we assume the processors are synchronous and preemptive, and the processing distribution P_{ii} is geometric with parameter $\mu_{ii} > \mu_c$, $i \in \mathbf{M}$. In the following proposition, we show that π_{MSR} outperforms π_{MWQ} with respect to average system size.

Proposition 4.1. *If $\vec{\lambda} \in \Lambda$, then*

$$E \left[\sum_{i=1}^M N_i^{\pi_{\text{MSR}}}(t) \mid N^{\pi_{\text{MSR}}}(0) \right] \leq E \left[\sum_{i=1}^M N_i^{\pi_{\text{MWQ}}}(t) \mid N^{\pi_{\text{MWQ}}}(0) \right] \quad (4.1)$$

for $t \geq 0$, provided $N_i^{\pi_{\text{MSR}}}(0) = N_i^{\pi_{\text{MWQ}}}(0)$, $i \in \mathbf{M}$.

Proof. The proposition is established via a standard interchange argument (see [13]). It suffices to show there exists a policy $\pi \in \mathbf{\Pi}$ which follows π_{MSR} at $t = 0$ (and is appropriately defined at all other decision instants), and is such that

$$E \left[\sum_{i=1}^M N_i^\pi(t) \mid N^\pi(0) \right] \leq E \left[\sum_{i=1}^M N_i^{\pi_{\text{MWQ}}}(t) \mid N^{\pi_{\text{MWQ}}}(0) \right] \quad (4.2)$$

for $t \geq 0$. By inductively applying the argument, we establish the proposition.

Without loss of generality, assume $N^\pi(0) = N^{\pi_{\text{MWQ}}}(0) = z$ and $z_i \geq M$, $i \in \mathbf{M}$. Suppose that the ℓ th queue initially contains the greatest number of jobs. Define $\mathcal{A}_0 = \{k \in \mathbf{M} : N_k(0) > 0 \text{ and } U_k^{\pi_{\text{MWQ}}}(0) \neq e_k\}$ to be the set of processors that, at $t = 0$, are not allocated by π_{MWQ} to the queues where they are most efficient; by definition, these processors must be allocated to the ℓ th queue at $t = 0$.

Couple the realizations of the queue length processes under π_{MWQ} and π as follows. Give the i th queue in each system the same arrival process, $i \in \mathbf{M}$. If $k \notin \mathcal{A}_0$, then during $[0, 1)$, the k th processor in each system is allocated to either the ℓ th queue or the k th queue, depending on whether the k th queue is initially empty or not. During $[0, \infty)$, let the k th processor under π take the same actions and experience the same service outcomes as the k th processor under π_{MWQ} , $k \notin \mathcal{A}_0$.

If $k \in \mathcal{A}_0$, then during $[0, 1)$, the k th processor is allocated to the ℓ th queue under π_{MWQ} , and to the k th queue under π . Define $\eta_k > 0$ to be the first time that the k th processor in the system operating under π_{MWQ} is allocated to the k th queue, $k \in \mathcal{A}_0$. Under π_{MWQ} , if a class i processor, $i \neq j$, is allocated to a class j job at some time t , then all the class j processors must be allocated to class j jobs at t . Therefore η_k occurs before the first time that the k th queue empties, which occurs in finite time since π_{MWQ} is stable, $k \in \mathcal{A}_0$. During $[1, \eta_k)$ and $[\eta_k + 1, \infty)$, let the k th processor under π take the same actions and experience the same service outcomes as the k th processor under π_{MWQ} , $k \in \mathcal{A}_0$. At $t = \eta_k$, let π allocate the k th processor to the ℓ th queue. Give the k th processor in the system operating under π the service outcome during $[\eta_k, \eta_k + 1)$ that the k th processor in the system operating under π_{MWQ} experienced during $[0, 1)$, and vice versa, $k \in \mathcal{A}_0$. This shows that (4.2) holds, since $\mu_c \leq \mu_{jj}$, $j \in \mathbf{M}$.

If $z_i < M$ for some $i \in \mathbf{M}$, then by appropriately reallocating surplus processors in the event that more processors are allocated to a queue than there are jobs in that queue, it can be shown that (4.2) holds. \square

For a general processing system, the MSR policy has a smaller stability region than the MWQ policy (see also Remark at the end of Section 3). However, for a general two-queue, two-processor system with diagonally dominant \mathcal{M} service rate matrix, the stability regions of the two policies coincide and we can glean some insight about the relative merits of the two policies regarding average system issues. The system under consideration has geometric service times with rates $\mu_{11} = 0.8$, $\mu_{12} = 0.2$, $\mu_{21} = 0.3$, $\mu_{22} = 0.7$. In Fig. 1 the interpolated surfaces of the average backlogs for the two queues (calculated from 100 simulations) of the system are given. It can be seen that the MSR policy clearly outperforms the MWQ policy at the “corners” close to the axes of the stability region, where the almost uninterrupted allocation of the most efficient processor proves beneficial from a backlog point of view. Additional simulations for larger systems with a similar diagonally dominant service rate matrix confirm this finding. It can also be seen that for this small system the MSR policy is extremely competitive throughout the stability region.

Remark. For systems with complete load balancing, i.e. $\lambda_i = \lambda$ for all $i \in \mathbf{M}$, and in addition total symmetry with respect to their service structure, i.e. $\mu_{ii} = \mu$ and $\mu \geq \mu_c$, $i \in \mathbf{M}$ we can establish

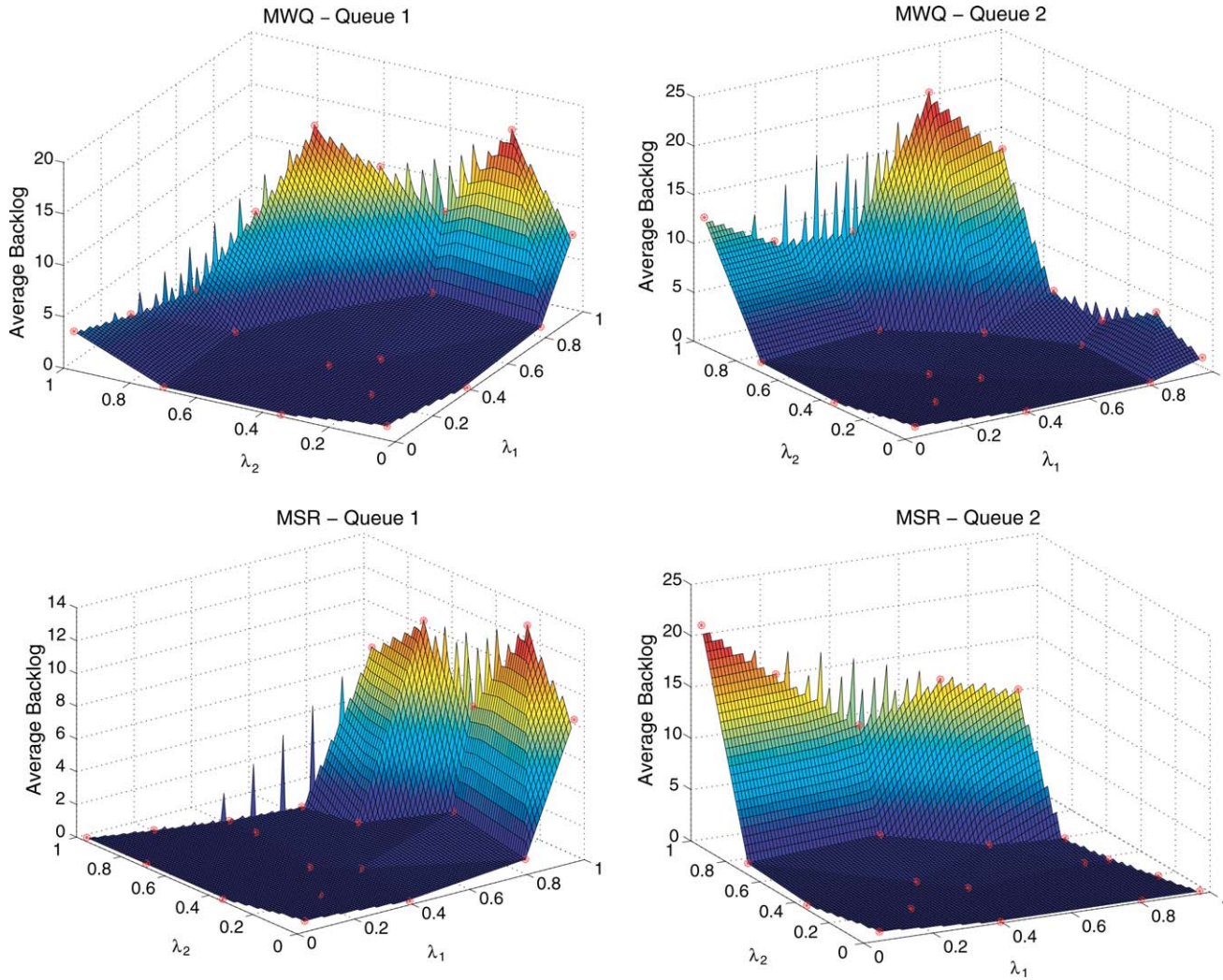


Fig. 1. Average backlogs for a two-queue, two-processor general processing system under the MWQ and MSR policies.

following analogous arguments as in Proposition 4.1, that whenever a class j processor is available, and the j th queue is empty, it is optimal to allocate it to the queue with the greatest number of jobs.

By exploiting the structure of π_{MSR} we can obtain a few simple (loose) bounds on the average system size. Define $B(p, q) = ((p/q)/(1 - p/q))(1 - p)$ for $p, q \in [0, 1]$ and $p/q < 1$. Fix $\vec{\lambda} \in \Lambda$ and assume that $\mathbf{N}^{\pi_{\text{MSR}}}$ is stationary. Let $\mathcal{G}_{\vec{\lambda}} = \mathbf{M} - \mathcal{B}_{\vec{\lambda}}$. Consider the policy π' defined as follows. Class $j \in \mathbf{M}$ processors give priority to class j jobs. If there are no class j jobs present in the system, then class j processors idle if $j \in \mathcal{B}_{\vec{\lambda}}$ and are allocated to class $\mathcal{B}_{\vec{\lambda}}$ jobs (but never to other class $\mathcal{G}_{\vec{\lambda}}$ jobs) if $j \notin \mathcal{B}_{\vec{\lambda}}$. It is clear that the number of class $\mathcal{B}_{\vec{\lambda}}$ ($\mathcal{G}_{\vec{\lambda}}$) jobs under π_{MSR} is stochastically lower (upper) bounded by the number under policy π' (due to the construction of this policy). This gives

$$E \left[\sum_{i \in \mathcal{B}_{\vec{\lambda}}} N_i^{\pi_{\text{MSR}}} \right] \geq B \left(\sum_{i \in \mathcal{B}_{\vec{\lambda}}} \lambda_i, \sum_{i \in \mathcal{B}_{\vec{\lambda}}} \mu_{ii} + \sum_{i \notin \mathcal{B}_{\vec{\lambda}}} \mu_c \left(1 - \frac{\lambda_i}{\mu_{ii}} \right) \right), \tag{4.3}$$

$$E \left[\sum_{i \notin \mathcal{B}_{\vec{\lambda}}} N_i^{\pi_{\text{MSR}}} \right] \leq \sum_{i \notin \mathcal{B}_{\vec{\lambda}}} B(\lambda_i, \mu_{ii}), \tag{4.4}$$

where we have dropped the dependence on t , since $\mathbf{N}^{\pi_{\text{MSR}}}$ is taken to be stationary. Now, turning to the general case, let $\mu^* = \max_{j \neq i} \{\mu_{ji}\}$ and assume $\mu_{ii} \geq \mu^*, i \in \mathbf{M}$. Then by stochastic dominance and Proposition 4.1, and assuming $\mathbf{N}^{\pi_{\text{MWQ}}}$ is stationary, we have

$$E \left[\sum_{i=1}^M N_i^{\pi_{\text{MWQ}}} \right] \geq E \left[\sum_{i \in \mathcal{B}_{\vec{\lambda}}} N_i^{\pi_{\text{MWQ}}} \right] \geq B \left(\sum_{i \in \mathcal{B}_{\vec{\lambda}}} \lambda_i, \sum_{i \in \mathcal{B}_{\vec{\lambda}}} \mu_{ii} + \sum_{i \notin \mathcal{B}_{\vec{\lambda}}} \mu^* \left(1 - \frac{\lambda_i}{\mu_{ii}} \right) \right). \tag{4.5}$$

5. Concluding remarks

In this paper, the problem of dynamic allocation of the resources of a general processing system comprised of varying efficiency processors is studied. It is shown that the MWQ policy that does not require knowledge of the system loading achieves maximum throughput under both preemptive and non-preemptive service disciplines. For systems with a certain degree of structural symmetry a second policy (MSR) is introduced and its performance characteristics both with respect to throughput and average backlogs is analyzed.

An interesting issue for further investigations is whether appropriate modifications of these policies continue to exhibit optimal behavior in the presence of finite buffers.

Acknowledgments

The authors would like to thank the AE and three anonymous referees for many comments and suggestions that substantially improved the presentation of the paper. Research supported in part by the National Science Foundation under grants IIS-9988095, DMS-0214171 and CCR-0325571.

References

- [1] M. Armony, N. Bambos, Queueing dynamics and maximal throughput scheduling in switched processing systems, *Queueing Syst.: Theory Appl.* 44 (2003) 209–252.
- [2] X. Du, X. Zhang, Coordinating parallel processes on networks of workstations, *J. Parallel Distribut. Comput.* 46 (1997) 125–135.
- [3] B. Hajek, Hitting times and occupation-time bounds implied by drift analysis with applications, *Adv. Appl. Probabil.* 14 (1982) 502–525.
- [4] W. Lin, P.R. Kumar, Optimal control of a queueing system with two heterogeneous servers, *IEEE Trans. Automat. Control* 29 (1984) 696–703.
- [5] T. Lindvall, *Lectures on the Coupling Method*, Wiley, New York, 1993.
- [6] P. Luh, I. Viniotis, Optimality of threshold policies for heterogeneous server systems, in: *Proceedings of the 29th Conference on Decision and Control*, Honolulu, HI, 1990.
- [7] L.M. Ni, P.K. McKinley, A survey of wormhole routing techniques in direct networks, *IEEE Comput.* 26 (2) (1993) 62–76.
- [8] R. Pemantle, J.S. Rosenthal, Moment conditions for a sequence with negative drift to be uniformly bounded in L^r , *Stochastic Process. Appl.* 82 (1999) 143–155.
- [9] R. Righter, Loading and sequencing on parallel machines, *Probabil. Eng. Inform. Sci.* 6 (1992) 193–199.
- [10] S. Ross, *Stochastic Processes*, Wiley, New York, 1983.
- [11] L. Tassiulas, A. Ephremides, Stability properties of constrained queueing systems and scheduling policies for maximum throughput in multihop radio networks, *IEEE Trans. Automat. Control* 37 (1992) 1936–1948.
- [12] J. Walrand, A note on optimal control of a queueing system with two heterogeneous servers, *Syst. Control Lett.* 4 (1984) 131–134.
- [13] J. Walrand, *An Introduction to Queueing Networks*, Prentice-Hall, Englewood Cliffs, 1988.
- [14] K.M. Wasserman, G. Michailidis, N. Bambos, Differentiated Processors' Scheduling on Heterogeneous Task Flows, Technical Report SU-NetLab-2001-12/1, Engineering Library, Stanford University, CA, 2001.
- [15] R.R. Weber, P. Varaiya, J. Walrand, Scheduling jobs with stochastic processing requirements on parallel machines to minimize makespan or flowtime, *J. Appl. Probabil.* 23 (1986) 841–847.
- [16] G. Weiss, M. Pinedo, Scheduling tasks with exponential service times on non-identical processors to minimize various cost functions, *J. Appl. Probabil.* 17 (1980) 187–202.